RecycleBot

Meghana Keeta Author, Serena Ying Author, and Mae Zhang Author

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— Plastic waste has been a major focus of environmental awareness in recent years. In order to combat this issue, we have designed a RecycleBot to autonomously detect and collect plastic bottles into its storage for later recycling. RecycleBot uses computer vision and machine learning to efficiently detect, navigate to, and collect plastic bottles into its storage for later recycling. It should be able to collect plastic bottles of various shapes, orientations, and sizes and also avoid obstacles in its path. RecycleBot not only removes the need for people to manually pick up littered bottles, but also improves the rate of recycling such items, contributing to the effort to preserve our planet.**

***Index Terms*—computer vision image processing, iRobot Create 2, LiDAR depth sensing, machine learning object detection**

I. Introduction

22 billion plastic water bottles get thrown away rather than recycled every year in the US [1]. Furthermore, more than 60 million plastic water bottles are thrown away each day, many of which end up as litter in streets, parks, and waterways [2]. It is evident that plastic bottles are a critical source of litter in the US, which means that the endeavor to reduce the litter in public areas is high effort in terms of resources and manpower.

RecycleBot can autonomously travel within an area to systematically scan for and detect plastic water bottles in an environment with litter, then travel to the location of the detected bottle and collect it into its own storage for later recycling. Users of RecycleBot will benefit twofold from its autonomous bottle collection functionality— on one hand, removing the need for human labor to reduce the amount of plastic litter, and on the other, automating the process of sorting between recyclable plastic water bottles and other types of non water bottle or non-recyclable waste. Hence, this solution will not only reduce the amount of resources dedicated towards managing plastic bottle litter, but also improve the rate of recycling such items, for both economic and environmental gain.

Research into competing and adjacent technologies reveals that automated recycling systems are available, however, many aim to automate the sorting process between recycling and trash [3], which still requires humans to deposit their trash into designated areas. Other mobile cleaning robot solutions include those meant to collect litter in local waterways [4] or litter on beaches [5]. Beyond the location difference between such robots and RecycleBot which is meant to operate on hard, concrete surfaces, these robots are more focused on collecting trash rather than collecting only recyclable items which can all go into recycling without additional sorting.

II. Use-Case Requirements

To ensure RecycleBot meets its intended goal to efficiently detect and collect plastic bottles for recycling, we have the following use case requirements.

RecycleBot must be **fully autonomous and operate on smooth, hard terrain**. The RecycleBot being fully autonomous is crucial to its success, as we are aiming to reduce the manpower required to clean up public spaces. Our design will be meant for cleaning public places with hard terrain because it is the most feasible and useful area of operation.

In order to collect bottles successfully, they will first need to be identified in the environment. RecycleBot's bottle detection model needs to correctly identify the following items with less than a 10% false positive rate:

1) Fixed-type water bottles with 90% success
2) Varied-type water bottles with 80% success
3) Obstacles with 80% success

In this paper, fixed-type water bottles refer to bottles that are knocked over, standard size, and not crushed. Varied-type water bottles refer to bottles of different sizes, shapes, colors, and orientations. Obstacles are anything in the environment that is not a bottle. These can be other commonly littered items such as aluminum cans and paper. These classification accuracies will be tested in the real world environment where there will be items scattered around the RecycleBot within a 1 meter radius. We chose **90% accuracy for fixed-type bottle detection** because the detection model may misclassify on instances where the bottle is too far away or obscured by another item. We chose **80% accuracy for varied-type bottles and obstacles**, which is a drop in accuracy compared to the 90% expectation for fixed-type bottles, because of the wide range of variation within these two categories, which will be more difficult for our model to classify.

Next, we require RecycleBot to **avoid obstacles with a 80% success rate**. In the real world, we expect items that are not bottles in the environment, so RecycleBot needs to be able to successfully avoid bumping into or picking up obstacles that are in its path. We have allowed for a margin of error due to the fact that the robot may have complications picking up accurate distance readings and stopping in time while operating at an efficient speed.

To confirm successful navigation and hardware mechanics of the system, we require RecycleBot to **pick up and store detected plastic water bottles with a 70% success rate**. This tests the RecycleBot's ability to successfully navigate to a detected bottle and activate its intake to collect the object into its internal storage area. We have allowed for a margin of error due to the various sizes, shapes, and orientations of bottles the intake is expected to pick up.

Finally, we require RecycleBot to be timely and efficient. RecycleBot is required to take **on average less than 40**

seconds to identify an attempt to pick up 3 items distributed within a 1 meter radius with no obstacles present. To elaborate on the quantitative details of this task, we referenced a real world use case of RecycleBot to determine both the timing and the search area. As RecycleBot is expected to be used on smooth concrete, we chose a basketball court to be our test field, given their relative ubiquity in public parks and likelihood to be littered with plastic water bottles and other types of waste. For our search area, since we want RecycleBot to search in circular areas each run, eventually covering the entirety of the basketball court, we had to consider the way that RecycleBot would be able to navigate to a new point to center its search area in between runs. The most logical way would be to utilize GPS to pinpoint RecycleBot's location and to calculate the location for the new center point. Through research into GPS accuracy limitations, we found that in practice, the best case horizontal error of GPS assisted by the Wide Area Augmentation System (WAAS) is 1 meter [6]. However, to account for outdoor conditions where there is less visible sky and thus a reduced accuracy of received location data from the WAAS system of satellites, we assume a worst case horizontal error of 2 meters. Thus, we want our robot to see a maximum of 2 meters in any direction, which is why we chose a search radius for this test of 1 meter. For our timing requirement, we want RecycleBot to be able to clear an entire basketball court in an hour in order to avoid extreme light conditions changes such as the sun setting affecting the visibility of the robot, since many public parks in the US close near sunset and would likely run RecycleBot near closing time. With a search radius of 1 meter, RecycleBot would need to complete around 100 iterations to cover a basketball court. To complete its searches in an hour, it would need to finish each run in 36 seconds on average. We have rounded this value up to 40 seconds since we don't expect to see such a high density of water bottles on the court (3 bottles in 1 m radius).

III. Architecture and/or Principle of Operation

Our system relies on computer vision and liDAR to detect and locate plastic bottles and obstacles, then sending navigation commands to the iRobot and intake mechanism. These system components are outlined in Figure 1.

From the environment, the RealSense sends real time image color streams and image depth streams to the Jetson Xavier to do all the software processing. Software processing includes inference with a trained neural network, target selection, object tracking, and navigation calculations. Navigation calculations include the angle to turn to center the target to the robot and distance between the robot and target. These navigation commands are sent to the iRobot over UART and the Arduino Uno which controls the motor driver for the intake mechanism.

A. Robot structure

The hardware and robot components are shown in the CAD model in Figure 2 and Figure 3. The chassis of our robot that allows the robot to move is an iRobot. We will be using a UART cable to connect the iRobot to the Jetson. Then we are building all the other components on top of and around the iRobot. To mount the other components to the iRobot we will be using Wafer screws that are screwed from the bottom of the iRobot nameplate to the wood on top. The iRobot only has top mounting through the nameplate, so the rest of the structure will need to be supported by the wood beam in the center. We believe the wood will be sturdy enough for this to be possible along with the rest of the system. The center wood beam will be connected to 2 side wood beams. These wood beams will be leaning against the sides of the iRobot and the ground. Connected to the center wood beam
and the side wood beams will be a lexan sheet that goes on top of the iRobot. This will be the bottom of the water bottle storage area.

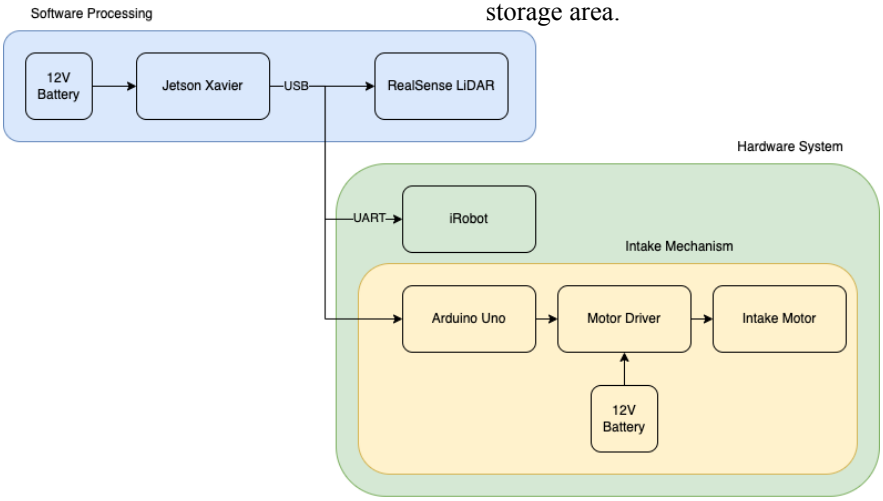

Figure 1: System diagram

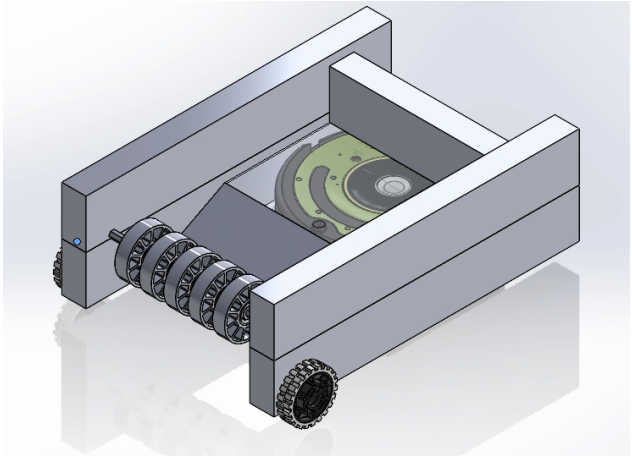

Figure 2: Robot CAD Isometric View

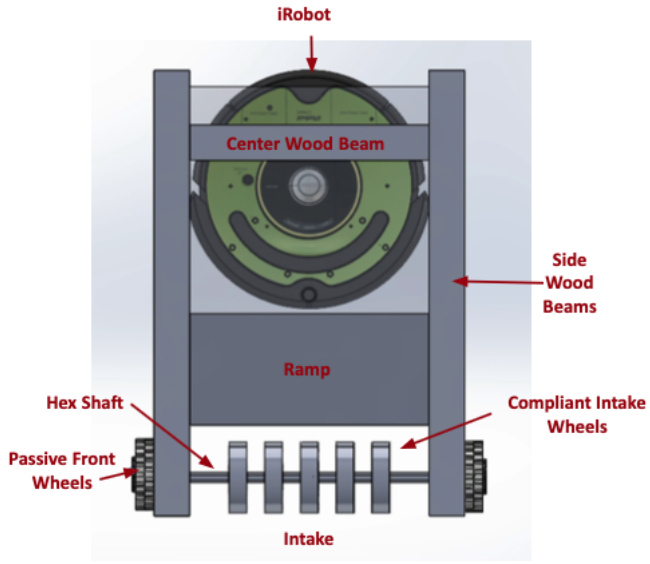

Figure 3: Robot CAD/Labeled Robot Parts Top View

B. Intake

In the front of the robot, the intake will be mounted. There will be intake mounting plates that are screwed into the side wood pieces on both sides (these are abstracted away in the CAD model). The intake mount plate will hold our intake. There will be a ½ inch hex shaft that will hold the wheels for the intake. The wheels are green compliant wheels with a 35A durometer. We will use spacers between the wheels so they are held in place. Then at the ends of the wheels we will use ½ inch shaft collars to cheap the wheels and spacers from moving. The entire shaft will be connected to the motor plates using more shaft collars and bearings to allow the shaft to rotate. The shaft will be directly connected to the motor, which will be mounted on the side (abstracted away in the CAD drawing). A labeled photo of the intake is shown in Figure 4 below.

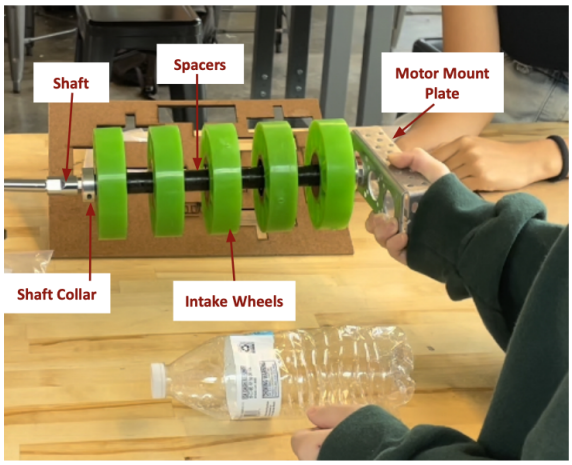

Figure 4: Robot Intake

C. Electrical components

Since the driving for the robot is controlled by the iRobot, the only electrical components for the robot we needed to figure out is for the intake. The intake will have one motor that powers it from one side. The motor will be connected to a motor driver, which will be connected to an Arduino. The Arduino will be connected to the Jetson Xavier to be integrated with the Intel Realsense and the object detection algorithm.

D. Camera and component mounting

All the electrical components and wiring will be done on top of a lexan plate. The lexan plate will sit on top of the side wood beams. The camera will be mounted in the middle of this lexan plate. The other components mounted there include the Nvidia Jetson Xavier, Arduino Uno, and motor driver. The wiring between all these components will also be housed there.

IV. Design Requirements

Our design requirements define the expected technical behaviors of our system and relate to our use case requirements.

For our software system, we impose a requirement for the FPS process rate of the image stream from the LiDAR camera. Through testing, we know that the bottleneck in the software system is running inference on the sampled image stream, so we hope to run inference at 1 frame per second in order to minimize lag. This means our trained ML model will only output bounding boxes around detected objects every second. Since this rate limits the amount of information the robot has about its environment if solely reading from inference, we use an openCV tracking algorithm on 10 frames per second in order to bridge the frame gaps in between each frame we sample for our trained ML model. Since the openCV tracking algorithm works well in real time, we plan to utilize the tradeoff between processing rate and number of tracked

objects in order to track all the detected objects in the robot's field of view, allowing us to not only maintain accuracy when traveling towards a bottle but also to detect and avoid obstacles. Including the supplemental use of a tracking algorithm to follow the labeled bounding boxes given by our model in between allows RecycleBot to still collect sufficient data to detect bottles and obstacles in its field of view without being bottlenecked by how fast inference takes.

Another requirement for our software system involves the range at which we want our robot to be able to classify items accurately. Since our use case requirements define the robot's task of picking up 3 bottles within a 1 meter radius, we want our robot's bottle detection algorithm to be able to accurately identify and label objects from a maximum distance of 2 meters away. In testing, we have discovered that our current implementation may need enhancing to reach this vision benchmark, so we plan to append to our training dataset so that inference works better on objects further away or doing preprocessing on the raw images such as sectioning the raw image into multiple zoomed in partitions to be passed through the trained model instead.

As per our use case requirement of picking up 3 bottles within 1 meter, we assert a hardware requirement that RecycleBot should hold 3 bottles in its on-unit storage. Since empty plastic water bottles weigh around 20 grams each, the additional weight when carrying 3 bottles in its storage is largely negligible. The main factor limiting the amount of bottles RecycleBot can collect is its storage capacity, an area around 25 square centimeters. This should hold approximately 3 bottes when accounting for differing shapes and crush levels.

We additionally have a hardware requirement of the robot being able to complete 10 runs of the 1 meter test consecutively. As a part of our testing structure, we will be deriving averages of each task over 10 runs, and thus should run the robot through each task 10 times in a row without any components losing power.

V. Design Trade Studies

A. *Software*

- **Yolov5**: Yolov5 [7] is a popular model architecture and algorithm for object detection. We chose to fine tune a Yolov5 model because it maintains accuracy and is computationally fast compared to other models.
- **Drinking waste dataset:** To train our model for object detection, we researched many open source datasets and found the most suitable one [8]. This dataset contains various images of plastic bottles, aluminum cans, glass bottles, and milk bottles. Since it has images of plastic bottles of different shapes, sizes, and orientations and other drinking waste to be classified into as obstacles, we conclude that this data is suitable for our use case. It has annotations indicating the bounding box around each object and its label, which is compatible with Yolo. This saves us the work of making our own dataset from scratch.
- **CSRT tracking algorithm**: We chose to include a tracking algorithm in our software pipeline in order to increase the number of FPS processed for the robot to properly move toward the target and avoid obstacles. Rather than performing inference on every sampled frame while the robot is moving towards the target, the object tracking would be able to trace the positions of the objects found during inference. We chose this particular tracking algorithm out of the 8 total tracking algorithms housed in the openCV library due to its tendency to display a high tracking accuracy [9]. The CSRT Tracker, which is a Discriminative Correlation Filter (with Channel and Spatial Reliability) based object tracker, is aided by spatial reliability maps that manipulate and select the filter with the highest quality for application to the tracked object's region, which increases the search area and includes robustness to track non-rectangular objects, however at the expense of a slower throughput [10]. Selecting a more accurate tracking algorithm that was also able to track the target even when other objects overlap was critical to the use case of RecycleBot, as we wanted to ensure that our system would not lose track of the target plastic bottle as it moved towards it. Since we would already be sampling the camera stream rather than processing it at 30FPS, which is the depth frame rate of the L515, we decided CSRT's slower FPS throughput would be a nonissue.
- **Rotation**: We decided to have RecycleBot rotate 60 degrees clockwise at the end of its detection cycle, when it fails to identify a plastic bottle within its field of vision. The motivation behind this decision comes from the range of vision of the RealSense L515, which has a depth field of view of 70 degree by 55 degree (+/- 3 degrees). Rotating 60 degrees in order to search in a new field of vision not only allows for a discrete number of turns to cover the full 360 degrees around the robot, but also introduces some redundancy at peripherals of L515 field of vision in case the bottle detection algorithm fails to identify a bottle at the edge of vision.

B. *Hardware/Robotics*

- **LiDAR camera L515**: We chose to use a LiDAR camera to augment our object detection algorithm, in order to implement a more effective navigation system. Without the depth points generated by the LiDAR camera, we would have to manually calculate the distance that RecycleBot would need to travel based purely on the robot speed and the relative change in size of the detected

object, which would introduce margins of error due to noise and tracking loss. Since the only RealSense camera available from ECE lending was the L515, and purchasing a D400 series RealSense at $200+ price points was out of our budget given the sum of the materials that would needed to construct the intake and support on the iRobot, we decided to go with the L515 as our vision system. Since we will test indoors for practicality, we concluded that the L515 would be sufficient. The L515 provides a horizontal depth field of vision of 70 degrees, which is large enough to perceive a significant area within the 1 meter radius, as well as a depth frame rate of 30 FPS, which is more than enough for our software pipeline, as we will be sampling images at a minimum rate of 10 FPS. The minimum distance to receive depth readings is around 25 cm, and with the projected distance between the camera mount and the intake mechanism being greater than that distance, the camera will be able to provide depth readings for the bottles even as they are swept up into the storage area. It also records at 2 MP, which provides sufficient color resolution to utilize an openCV tracking algorithm effectively. While a LiDAR camera is more suited for an indoor environment due to its laser scanning technology, we've designed our software using the Realsense SDK, which is compatible with all camera series in the RealSense line, including other cameras that showcase better performance in outdoor environments [11]. Therefore, the project can be modified when scaled to use a more appropriate camera.

- **Jetson Xavier**: In our conversations deciding what we wanted as the brains of the system, we considered Jetsons and RaspberryPis. However, we quickly discovered that for running multiple specialized tasks such as our computer vision and machine learning requirements, the unit most capable of, suited to, and documented for the computing demands we needed was a Jetson. With the knowledge that NVIDIA Jetsons are used for computer vision and neural network applications including image classification and object detection, we initially planned on utilizing the Jetson Nano, which has a 4-core CPU and 128-core Maxwell GPU with 4GB of memory, and runs on 5 Watts [12]. Its small size and low power consumption were two of our initial motivations for choosing it as our computing element. However, after considering other options in the Jetson line, combined with the discovery of the availability of a Jetson Xavier from ECE lending, we switched to using a Jetson Xavier NX. With a 6-core CPU and 384-core NVIDIA GPU with 8GB of memory, running on 10W, the Jetson Xavier NX holds far more processing power but is still housed on a unit the same size as the Nano [13]. Given its computing power advantage over the Nano, we concluded that the Xavier would be the best choice for our computer, given its size, low power consumption, and more extensive CPU and GPU. These features, combined with its pre-installed openCV, made it the most logical choice for our use case.
- **iRobot**: We pivoted from planning to build the entire robot from a Rev Robotics kit to working with an iRobot Create 2 instead to cut down on the amount of time spent on building robot chassis that would take away from our time allocation for developing the intake mechanism. This tradeoff not only reduced build time but also effort required to implement a motor control system—iRobot has a very well documented Open Interface for control through UART.
- **Rotating intake**: We spent several weeks working through the design of the rotating intake, iterating through several implementations before settling on an axle with rotating rubber wheels. One suggested method was a gate collection method, where a rising and falling gate would close to capture bottles that roll into the ground level intake opening. However, since this implementation would cause issues with the bottles potentially falling out of the storage, we went back to a previous idea with a rotating intake because it requires less additional structure and the intake itself blocks the collected bottles from rolling out. Through testing we have determined that the rotating intake will require sufficient speed to spin the bottles in the correct direction, rather than bumping it away from the intake. We will be testing the optimal speed and distance from the ground in the coming weeks. The actual material used to build the intake was decided based on what was readily available on popular robotics parts websites (AndyMark, Rev Robotics) and if the part sizes worked together well with the rest of the system. There are many complaint wheels with different durometers, but we chose the ones that are the most rubber-like and squishy. These qualities are better for an intake because the wheels will be able to conform around whatever is being intaked when the motor is spinning fast. There are also many shafts and wheel holes available and we chose a ½ inch hex because many of the other components were compatible with ½ inch hex such as the motor.
- **Robot construction materials:** The center and side beams could have been built from wood or a metal such as aluminum. Although aluminum is slightly lighter and usually comes with more premade mounting holes, we decided to go with wood. This is because wood provides more flexibility with design because it is easier to work with. We don't know exactly how long the side beams should be because we don't know the correct angle of the

ramp that will allow the water bottle to easily go up it. Because this unknown will only be resolved from prototyping, the side beam length is a little variable. Furthermore to cut aluminum we would need something like a CNC machine, versus a simple bandsaw for wood.

VI. System Implementation

A full software system diagram can be found at the end of the document in Figure 6 of the Appendix. We will go into the details of this implementation in this section.

A. ML model and training before runtime

To get data in the format to train and validate our model, we parsed our Drinking Waste Classification dataset [8] so that plastic bottles are in the bottle class and all other items are in the obstacle class. Then we split 80% of images to be in training and 20% in validation, making sure to randomize and that both training and validation have equal number of bottle and non-bottle images. We then did transfer learning on a pre-trained Yolov5 model, freezing all the backbone layers. We chose to do transfer learning because of the relatively small size of our dataset, which is approximately 4000 images. After experimenting with different hyperparameters, we ended up training for 10 epochs on the Yolov5s model, then fine tuning for 10 epochs with the default hyperparameters in the hyp.VOC.yaml in the Yolov5 library. When given images from the Drinking Waste Classification dataset, The model classifies bottles with 99% accuracy and obstacles with 97% accuracy. If real-time inference does not perform well, we may need to append a wider variety of images to the training dataset or adjust model hyperparameters.

B. CV & LIDAR at runtime

- **Depth Sensing with Intel RealSense LiDAR Depth Camera L515**: There are several uses for the LiDAR camera in our project. The main goals for the LiDAR camera are to collect depth readings for the detected bottles in order to determine which bottle is closest to the robot, as the RecycleBot will navigate to the closest bottle within its field of view. We anticipate taking an initial depth reading from RecycleBot's initial position, then continue to take readings as the bot moves towards the bottle in order to maintain an appropriate speed, and to stop a feasible distance away from the bottle in order to pick it up. The LiDAR camera also gives distance readings to avoid obstacles. We chose to use the Python wrapper library for the RealSense SDK to interface with the LiDAR Camera due to our familiarity with openCV and Python based machine learning and computer vision applications.
- **OpenCV Object Tracking**: The main purpose of using an openCV tracking algorithm (we chose CSRT) is to track bottles and obstacles when the robot moves towards a target, as well as to assist with rotation when RecycleBot needs to turn to face a target water bottle. As a part of our navigation algorithm, once RecycleBot has selected a target water bottle to attempt to pick up, it will first rotate to face the target head on. This entails first having to determine which side the bottle is located in, relative to the vertical center line within the bot's field of vision. Once the side is determined, we plan to send commands to the iRobot to begin to incrementally turn in that direction, and use openCV object tracking to continue following the selected object as its position changes within the robot's field of vision, and then to send the command to the iRobot to cease its turning when the center point of the identified bounding box around the target object is aligned with the vertical center line within the bottle's field of vision. Once the robot is aligned with the target, we will send commands to the iRobot to move it forward and continue to use object tracking and liDAR to see if there are any obstacles in the way.

C. Hardware and robotics at runtime

- **Power:** The components of our system which require power are the Jetson Xavier, the intake mechanism DC gearmotor, and the iRobot. Since the iRobot has a charging dock as well as an on-unit battery lasting 2 hours per charge, we did not need to provide extra power for it. Hence, we ordered a 11.1V 6000mAh power supply for the Jetson, which requires 12V and draws up to 15W of power, so on a single charge the power supply will sustain the Jetson for around 4 hours, which will be more than enough for our use cases. We also purchased 12V 23A batteries for powering the motor driver for the intake mechanism DC gearmotor, to be swapped out as needed.
- **Intake Mechanism:** The Arduino Uno controlling the motor driver for the intake will only allow the intake to spin when RecycleBot is within a certain distance from the target bottle.
- **iRobot Control:** The software processing will be done continuously with a rate of at least 10 frames per second so that the course and speed of the iRobot can be appropriately adjusted as it nears a bottle or reaches an obstacle. If it is nearing a bottle, it will slow down to allow the intake to collect the bottle and push it up a ramp into the storage. If it is nearing an obstacle, it should stop, turn 60 degrees clockwise, move forward .3m, and then restart the searching algorithm.

VII. Test, Verification and Validation

For testing, we have fixed some conditions in the environment:

1. **Objects will be randomly placed within a 1.5 meter radius**. We expect our computer vision to

detect items up to 3 meters away and a lack of GPS in our system constraints RecycleBot to only cover this amount of area. With a GPS, we can compose any grid of points 3 meters away from each other. For each point, we can use the same software to detect bottles and therefore, RecycleBot can be scalable with GPS.

2. **Objects will be at least .45 meters apart from each other**. This condition allows the robot to have a chance to successfully avoid obstacles. If an obstacle and bottle are placed too close to each other, our robot intake will not be able to pick up a bottle while avoiding the obstacle. We also do not expect bottles and obstacles to be so closely clustered in the real world.
3. **RecycleBot will operate on a concrete background**. We want to minimize our scope to only concrete background in accordance with our use case requirements.
4. **RecycleBot will operate under fixed lighting conditions**. Our detection model training dataset was under fixed lighting conditions, so the performance with darker lighting is unknown.

To evaluate our design, we have constructed 3 test cases, with all test cases operating under the fixed conditions listed above. These test cases only differ in the items placed within the 1.5 meter radius:

1. 3 fixed-type bottles
2. 3 variable-type bottles
3. 3 fixed-type bottles and 3 obstacles

We define one run to be the completion of one 1.5 meter radius. Since we have a small sample size of 3 bottles per run, we will compare the average of 10 runs with our metrics. For each test case, we are evaluating multiple metrics indicated in Table 1).

Testing	Metrics- all percentages are an average of 10 runs
1. 3 fixed-type bottles	90% bottle detection accuracy 70% pickup success 80% completion in less than 75 seconds
2. 3 variable-type bottles	80% bottle detection accuracy 70% pickup success 80% completion in less than 75 seconds
3. 3 fixed-type bottles and 3 obstacles	90% bottle detection accuracy 80% obstacle detection accuracy 80% obstacle avoidance success 70% pickup success

Table 1: Tests with corresponding metrics

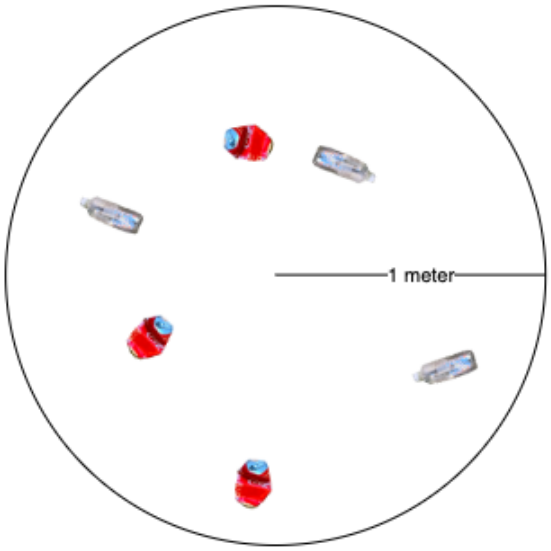

Figure 5: Visualization of test environment, specifically test 3

We wish to evaluate bottle detection accuracy in all three cases, as it is the crucial initial part of the pipeline. We will identify a bottle as detected if inference produces a bounding box around the bottle at any point during the test. The third test case specifically tests for obstacles by including them in the environment. We will identify an obstacle as detected in the same way as we do for bottles. For obstacle avoidance, we will consider only cases where the obstacle is blocking the path of the robot. Out of those cases, the success condition is if the robot stops and rotates before hitting the obstacle. We evaluate pickup success in all three tests, where a success consists of the intake pushing the item into the robot's storage. Speed is only evaluated in the first two test cases, as obstacles would likely introduce too much variance in our test data.

Each metric will be evaluated independently and we will allow the robot to keep running until it reaches its stop state. For instance, if we are calculating results of test 1 (3 fixed-type bottles) and the model only detected 20/30 (66%) of the bottles, we have failed the 90% bottle detection accuracy requirement. But if the robot has reached its stop state in less than 75 seconds in 8/10 runs, we have satisfied the speed requirement.

VIII. Project Management

A. Schedule

See Figure 7 for the full schedule on page 11.

B. Team Member Responsibilities

Team Member	Responsibilities
Meghana	- Designing and constructing an intake that collects water bottles from the ground and storing 3 bottles internally - Figuring out iRobot controls and motor controls for robot to move and collect bottles
Serena	- Working with the Jetson and Intel Realsense to achieve distance perception - Working on path and angle calculations for iRobot to travel
Mae	- Gathering the dataset, setting up and training a model to detect bottles and obstacles - Write logic for the software system

Table 2: Team member responsibilities

C. Bill of Materials and Budget

See Figure 8 in the Appendix Section on page 12 for the detailed bill of materials and budget. On a high level we are building off of an iRobot Create 2. We are using wood to connect the iRobot to our intake. We are also using aluminum and lexan for the ramp and storage area. On the hardware side, we are using an NVIDIA Jetson Xavier, Arduino Uno, a motor driver, and Intel Realsense. The other intake and construction materials have been purchased from Andy Mark, Amazon, Home Depot and GoBuilda.

D. Risk Mitigation Plans

The main risk that we will face is constructing the physical robot to match our use case requirements. Meghana has experience with similar robotics from high school, but hasn't built off of something like an iRobot before. The main worry with the iRobot is if it will be able to carry the wood, ramp and intake that is built on it and still move at a reasonable speed. Based on internet forums it seems like the maximum weight the iRobot can hold and still function normally is 15-20lbs [14]. We have picked out our build materials carefully to try to minimize this weight, but we won't know for sure how the robot will behave until you attach all the parts to the iRobot. If the construction is too heavy for the iRobot to function properly, we will minimize our design even further by either changing the location of the storage area or removing the storage area entirely.

Another similar robot related unknown is if the nameplate on top of the iRobot Create is strong enough to hold the wood. We researched the recommended way to mount attachments, but we believe the amount of stuff we are attaching is still a risk. If the nameplate mounting isn't strong enough, we will consider also using some sort of mounting on the sides of the iRobot.

A software related risk is if our robot will be able to detect water bottles at least 2 meters away. We know that the detection works close by, but we need to test if it works further because we need the robot to see at least this distance in order to satisfy our use case requirements. If the robot is not able to detect the bottle we will try appending more data to the training data, doing preprocessing on the image in realtime and zooming in or trying to do some sort of noise/background cancellation.

Overall, we have built in slack time to mitigate these risks and refocus on implementation details if we need to.

IX. Related Work

A variety of trash and recycle sorting robots exist, one of which is TrashBot [3]. TrashBot is a trash bin that is able to distinguish between trash and recyclable items. After throwing an item into TrashBot, it uses metal and computer vision to determine whether or not an item is recyclable. Upon classification, it uses a system of trap doors to discard the item into the correct bin. Such mechanisms help in the autonomous classification of trash, but still require people to properly dispose of their waste and therefore, they do not solve the problem of reducing existing litter in public places.

Many autonomous cleaning robot solutions with computer vision exist in today's industry. ClearBot, a robot that detects and collects litter in local waterways, has an autonomous setting where it uses computer vision to locate and collect floating pieces of litter [4]. Another such autonomous cleaning robot is BeachBot, a land operated robot designed to pick up cigarette butts on beaches [5]. The bot uses artificial intelligence to distinguish cigarette butts in the sand, then picks them up with grippers, and into an internal storage. It is also designed to avoid obstacles. Such robots are very similar to RecycleBot but have two main differences. First, the locations of operation differ since RecycleBot is meant to operate on hard, concrete surfaces. Second, these robots collect trash whereas RecycleBot collects only recyclable items, reducing the need to sort items into trash and recycling after collection.

X. Summary

We are working to create an efficient robot that will detect and collect water bottles from the ground. The robot will be autonomous and collect the water bottles without assistance.

Stakeholders can simply leave the robot where they want and the robot will clean, allowing stakeholders to use their energy on different things.

We will be using an iRobot Create 2, spinning intake and computer vision for our implementation of this robust robot. One upcoming challenge with this implementation is not having full autonomy over the chassis/drivetrain of the robot for mounting and functionality. However, we believe our design and plans will allow us to successfully mitigate these risks and meet our use-case and design requirements.

So far we are making great progress and anticipate finishing all the features we want to implement. We have ordered all of our parts necessary, started prototyping and working on the object detection/computer vision algorithm.

Glossary of Acronyms

FPS - Frames per second
CSRT - Discriminative Correlation Filter with Channel and Spatial Reliability
LiDAR - Light Detection and Ranging
WAAS - Wide Area Augmentation System

References

[1] https://www.americanrivers.org/2018/01/five-common-things-found-river-cleanups/

[2] https://healthyhumanlife.com/blogs/news/plastic-water-bottle-pollution-plastic-bottles-end

[3] https://cleanrobotics.com/trashbot/

[4] https://www.intelligentliving.co/autonomous-robot-collects-garbage-from-waterways/

[5] https://www.businessinsider.in/tech/news/meet-beachbot-a-beach-rover-that-uses-ai-to-remove-cigarette-butts-from-beaches/articleshow/84759758.cms

[6] https://www.robotsforroboticists.com/gps/#:~:text=Wide%20Area%20Augmentation%20System%20(WAAS)&text=In%20practice%20WAAS%20assisted%20GPS,the%20sky%20you%20can%20see

[7] https://github.com/ultralytics/yolov5

[8] https://www.kaggle.com/datasets/arkadiyhacks/drinking-waste-classification

[9] https://pyimagesearch.com/2018/07/30/opencv-object-tracking

[10] https://broutonlab.com/blog/opencv-object-tracking

[11] https://www.intelrealsense.com/lidar-camera-l515/

[12] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[13] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/

[14] https://robotics.stackexchange.com/questions/10686/how-much-weight-can-the-irobot-carry

APPENDIX

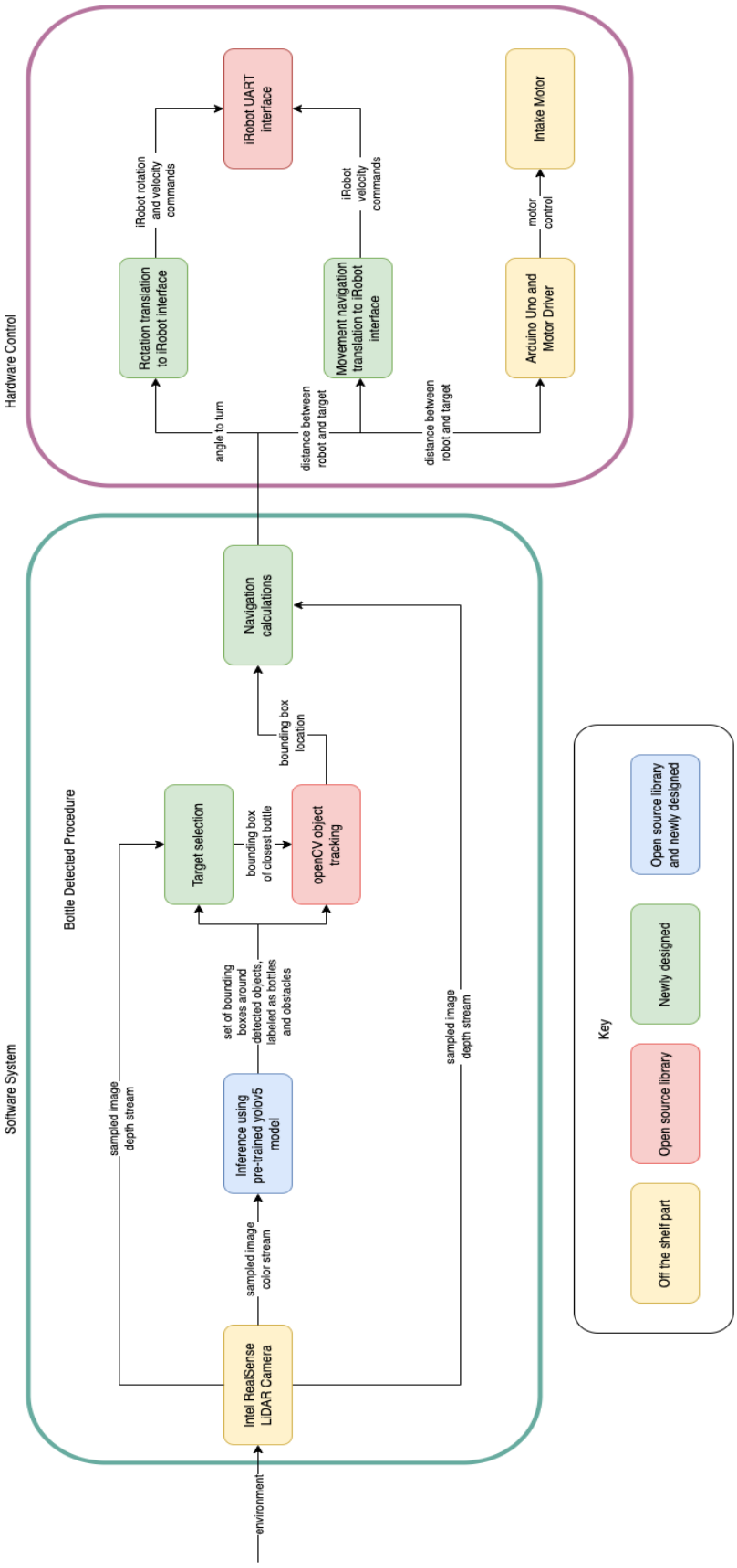

Figure 6: Software System Diagram

PROJECT	START DATE
Capstone Group A4: RecycleBot	Wednesday, September 14, 2022

TASKS	TASK OWNER	STATUS	START DATE	END DATE	DAYS
Deadlines					
Project proposal presentation	Serena	Complete	09/16/22	09/19/22	2
Design presentation	Mae	Complete	09/30/22	10/03/22	2
Final presentation	Meghana	Not Started	11/26/22	12/05/22	6
Object Detection and Identification					
Dataset generation	Mae	Complete	09/24/22	10/01/22	5
Get model running with initial dataset	Mae	In Progress	10/02/22	10/05/22	3
Train model to 90% test accuracy	Mae	Not Started	10/05/22	10/09/22	3
Integrate ML model into Jetson/detection process	Mae/Serena	Not Started	10/10/22	10/14/22	
Slack time	All	Not Started	10/17/22	10/25/22	7
Robot Construction and Software					
Research design implementations for using iRobot	Meghana	Complete	09/15/22	09/25/22	7
Research and design intake and storage	Meghana	Complete	09/20/22	09/26/22	5
Order Necessary Parts Materials	All	Complete	09/28/22	10/05/22	6
Robot CAD and Design	Meghana	Complete	10/02/22	10/07/22	5
Setting up Jetson to iRobot communication	Meghana	Complete	10/10/22	10/14/22	5
Prototyping Intake	Meghana	Complete	10/10/22	10/14/22	5
Constructing and Building robot	Meghana	In Progress	10/24/22	10/31/22	6
Controls to allow robot to intake items	Meghana	Not Started	10/24/22	10/31/22	6
Robot driving and intake testing	All	Not Started	10/31/22	11/04/22	5
Slack time	All	Not Started	11/05/22	12/02/22	20
Navigation and Integration					
Set up Jetson and Intel Realsense	Serena	In Progress	09/24/22	10/01/22	5
Distance perception with LIDAR	Serena	In Progress	10/02/22	10/09/22	7
Object tracking algorithm	Serena	In Progress	10/09/22	10/16/22	5
Path and angle calculation to motor movement	Serena	Not Started	10/22/22	10/25/22	2
Write logic for entire software system	Mae	Not Started	10/22/22	10/25/22	2
Put entire software system together	Mae/Serena	Not Started	10/25/22	10/31/22	5
Completely merge software and hardware systems	All	Not Started	11/01/22	11/08/22	6
Slack time	All	Not Started	11/09/22	11/19/22	8
Final testing					
Whole system to detect and pick up 1 water bottle	All	Not Started	11/20/22	11/28/22	6
Test system logic for picking up multiple bottles	All	Not Started	11/28/22	12/01/22	4
Test fixed bottle environment	All	Not Started	11/20/22	11/28/22	6
Test varied bottle environment	All	Not Started	11/28/22	12/01/22	4
Slack time	All	Not Started	12/01/22	12/04/22	2

Figure 7: Schedule

Figure 8: Bill of Parts

Item	Link	Cost Per Part	Quantity	
Intake Materials				
NeveRest Classic 60 Gearmotor	https://www.andymark.cor	16	1	16
NeveRest Classic Clamping moto	https://www.gobilda.com/1	7.99	2	15.98
Motor Driver Board Dual H Bridge	https://www.amazon.com/	11.97	1	11.97
4in Compliant Wheels Green 1/2 H	https://www.andymark.cor	9	5	45
0.5in Steel Hex Shaft 1 feet	https://www.andymark.cor	5.5	1	5.5
0.5 in Hex HD Split Collar Clamp	https://www.andymark.cor	3.75	2	7.5
Intake Motor Mount Material	https://www.andymark.cor	12	2	24
Hex Molded Spacers 1/2 Hex 0.25	https://www.andymark.cor	0.5	20	10
6mm D to Hex Adapter	https://www.andymark.cor	11	1	11
Intake Tubing Polyurethane	https://www.amazon.com	9.99	1	9.99
PVC Pipe 1/2 inch	https://www.homedepot.c	2.96	1	2.96
0.5 in Hex Bore 2 Piece Collar Cla	https://www.andymark.cor	7.5	4	30
0.5in Hex and Keyed Shaft Coupli	https://www.andymark.cor	9	1	9
1/2 in Hex Shielded Bearing	https://www.andymark.cor	6	2	12
Robot Base Construction				
Polycarbonate Sheets (11x14)	https://www.homedepot.c	10.98	2	21.96
Wood Stud 2inch x 4inch x 8ft	https://www.homedepot.c	3.98	1	3.98
Wood 2inch x 8inch x 8ft	https://www.homedepot.c	7.65	1	7.65
Aluminum 5052 12" x 12" sheet	https://www.amazon.com/	12.99	1	12.99
3/8" Metal Screws (25 pack)	https://www.homedepot.c	5.97	1	5.97
Wafer head screws #10, 1-1/2" - 5	https://www.amazon.com/	9.28	1	9.28
Power Supply (12V, 6A) for Jetsor	https://www.amazon.com/	39.79	1	39.79
12V Batteries for motor	https://www.amazon.com/	2.74	1	2.74
12V Battery Case for motor	https://www.amazon.com/	4.95	1	4.95
Arduino Uno	Already Own	0		0
Front Passive Wheels				0
1/2 Inch Wheel Axles	https://www.amazon.com/	8.5	1	8.5
1/2 inch Round Collar Clamp	https://www.andymark.cor	7.5	4	30
Hi Grip Wheels 4inch 80A	https://www.andymark.cor	4	2	8
1/2 inch Bearing	https://www.andymark.cor	1.5	4	6
			Total Cost	372.71